LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-772639

1D Transport Using Neural Nets, SN, and MC

M. M. Pozulp

April 23, 2019

M&C 2019
Portland, OR, United States
August 25, 2019 through August 29, 2019

Disclaimer

1D Transport Using Neural Nets, S_N, and MC

Michael M. Pozulp

***Abstract*—A method previously introduced to solve the discrete ordinates neutron transport equation (NTE) in a slab geometry using a neural network (NN) is revisited. A runtime comparison shows that the NN method can be faster than S_N and MC while achieving the same accuracy for an example problem.**

I. Introduction

THE field of *machine learning* (ML) applies mathematical models to data as a means of solving problems such as classification. The wide adoption of ML in areas ranging from consumer applications such as recommendation systems to basic science largely owes itself to the availability of large datasets and specialized computing hardware such as Google's tensor processing unit (TPU) [15] and NVIDIA's tensor cores [2]. A product of this success is advances in ML methodologies, as well as highly-tuned open-source software libraries such as `Scikit-learn` [13], `TensorFlow` [3], and `Torch` [12]. These advances can be applied to other technical problems, leading to further innovation. For example, Burgers' equation, Schrödinger's equation [14], and Laplace's equation [5] can each be posed as machine learning problems and solved using a function approximation technique known as a *neural network* (NN) [7]. In this work, these advances are leveraged to solve the neutron transport equation (NTE) in a slab geometry. The NTE describes the dynamics of the neutron population as neutrons move through material, undergoing nuclear reactions. The NTE has many applications, such as determining the output power of nuclear reactors, or the absorbed dose of ionizing radiation in medical patients.

Traditional numerical methods used for solving the NTE are either deterministic (e.g., finite element analysis) or stochastic (e.g., Monte Carlo). Deterministic methods suffer from errors introduced by discretization, and stochastic methods introduce statistical uncertainty, and both methods are computationally expensive. Here, a method previously introduced by [4] that uses NNs is explored. In particular, these methods are implemented using a modern deep learning framework and compared to traditional methods, including runtime comparisons, and is done so using a reproducible framework [1].

The remainder of this paper is outlined as follows. Section II introduces the slab geometry transport equation and methods for solving it using ANNs, the S_N method, and a Monte Carlo approach. Section III presents a quantitative comparison between the three methods. Lastly, Section IV discusses implications of the results and additional use cases that can be considered.

Michael M. Pozulp is with the Department of Computer Science at the University of California, Davis, CA 95616 USA and Lawrence Livermore National Laboratory, Livermore, CA 94550 USA. (email: pozulp1@llnl.gov)

II. Methods

A. Slab Geometry Neutron Transport

The general NTE is a function of seven variables: three spatial variables $\mathbf{r}$, energy E, direction $\hat{\Omega}$, and time t. Consider instead the slab geometry NTE, which is a function of a single spatial variable z and directional variable μ (see Appendix A for derivation)

$$\mu\frac{\partial\psi(z,\mu)}{\partial z}+\Sigma_t(z)\psi(z,\mu)=2\pi\int_{-1}^{1}d\mu'\Sigma_s(z,\mu_0)\psi(z,\mu')+\frac{1}{2}\left[\nu\Sigma_f(z)\phi(z)+Q_{ext}(z)\right] \quad (1)$$

where

- μ and μ' are the cosine of the polar angle of the direction vectors $\hat{\Omega}$ and $\hat{\Omega}'$, respectively, and $\mu_0 \equiv \cos\theta_0 \equiv \hat{\Omega}\cdot\hat{\Omega}'$
- ψ and ϕ are the angular and scalar neutron fluxes
- Σ_t, Σ_s, and Σ_f are the total, scattering, and fission macroscopic cross sections of the material, respectively
- ν is the neutron multiplicity
- Q_{ext} is an external source

Section II-B describes the use of a discrete ordinates approach to solve Equation (1), Section II-D describes the use of a neural network approach, and Section II-E describes a Monte Carlo approach.

B. Solutions using the Discrete Ordinates Method

The discrete ordinates (S_N) method discretizes the directional variable in the NTE, replacing the continuous directional variable with an angular quadrature set of order N. For the slab geometry in Equation (1), an even-order Gauss-Legendre quadrature set is employed. The spatial variable is discretized using the finite volume method, and a non-multiplying medium, meaning $\Sigma_f = 0$, is assumed. The resulting equations are (see Appendix B for derivation)

$$\frac{\mu_m}{l_j}\left(\psi^{(l+1)}_{m,j+\frac{1}{2}}-\psi^{(l+1)}_{m,j-\frac{1}{2}}\right)+\Sigma_{t,j}\psi^{(l+1)}_{m,j}=\frac{1}{2}\hat{Q}^{(l)}_{m,j} \quad (2)$$

where

$$\hat{Q}^{(l)}_{m,j}=\Sigma_{s0,j}\Phi^{(l)}_{0,j}+3\mu_m\Sigma_{s1,j}\Phi^{(l)}_{1,j}+Q_{m,j} \quad (3)$$

$$\Phi^{(l+1)}_{0,j}\equiv\sum_{m=1}^{N}\psi^{(l+1)}_{m,j}w_m \quad (4)$$

$$\Phi^{(l+1)}_{1,j}\equiv\sum_{m=1}^{N}\mu_m\psi^{(l+1)}_{m,j}w_m \quad (5)$$

$$\psi^{(l+1)}_{m,j}=\frac{1+\alpha_{m,j}}{2}\psi^{(l+1)}_{m,j+\frac{1}{2}}+\frac{1-\alpha_{m,j}}{2}\psi^{(l+1)}_{m,j+\frac{1}{2}} \quad (6)$$

with boundary conditions

$$\psi^{(l+1)}_{m,\frac{1}{2}} = f_m \qquad \mu_m > 0 \tag{7}$$

$$\psi^{(l+1)}_{m,J+\frac{1}{2}} = g_m \qquad \mu_m < 0 \tag{8}$$

In these equations,

- $m = 1, 2, \ldots, N$, $j = 1, 2, \ldots, J$, and $l = 1, 2, \ldots, L$ are the angular, spatial, and iteration indices, respectively
- $\mu_m \equiv \cos\theta_m$ is the cosine of the polar angle of the direction vector $\hat{\Omega}_m$
- l_j is the length of cell j
- $\psi^{(l+1)}_{m,j+\frac{1}{2}}$ is the cell-edge angular flux at iteration $l+1$
- $\Phi^{(l+1)}_{0,j}$ and $\Phi^{(l+1)}_{1,j}$ are the cell-averaged scalar flux and current at iteration $l+1$, respectively
- $\Sigma_{t,j}$, $\Sigma_{s0,j}$, and $\Sigma_{s1,j}$ are the total, total scattering, and linearly anisotropic scattering macroscopic cross sections of the material in the cell, respectively
- $Q_{m,j}$ is an external source

Using Equation (3), each spatial bin j is iterated through, updating the flux $\psi_{m,j+\frac{1}{2}}$ for each angular bin m. The bins are updated sequentially, starting from the left side of the slab to the right, then reverse order and repeat. The scalar flux in each bin j is updated until convergence, as determined by the relative pointwise convergence criterion

$$\max_{1\le j\le J}\left|\frac{\phi^{(l+1)}_j - \phi^{(l)}_j}{\phi^{(l)}_j}\right| < \epsilon \tag{9}$$

The S_N method is well understood [11], and is employed for production use in industry and academia.

C. Artificial Neural Networks

Artificial Neural Networks (ANNs) are a class of machine learning algorithms which provide a general means for non-linear function approximation. That is, ANNs can approximate an arbitrary function $f(\mathbf{x})$ for some input $\mathbf{x}$.

The general procedure for approximating functions using ANNs is to perform a series of operations on an input $\mathbf{x}$, which ultimately result in the output $f(\mathbf{x})$. The operations performed on the data $\mathbf{x}$ are specified by the user, and are collectively referred to as the *architecture* of the ANN. ANN architectures consist of *layers* which define the operations used. In general, ANNs consist of an input layer, one or more hidden layers, and an output layer. The term *deep learning* is a reference to this layer structure as deep learning architectures are often several layers deep. Each of these layers is connected in some way, and between them often exist *activation functions*, which may be nonlinear. For a layer to be *fully-connected*, each element of the layer must be connected to each element of the following layer.

Consider an ANN that takes as input $\mathbf{x} \in \mathbb{R}^d$ and has a single fully-connected layer with k nodes, and outputs $f(\mathbf{x}) \in \mathbb{R}^N$, as shown in Fig. 1. The connections between subsequent layers is represented as a matrix product between values of nodes and parameters of the network. In this example, the value at node i, v_i, is calculated as $v_i = \mathbf{w}_i\mathbf{x} + b_i$, where $\mathbf{w}_i$ is a vector of *weights* and b_i is a *bias*. The value at each

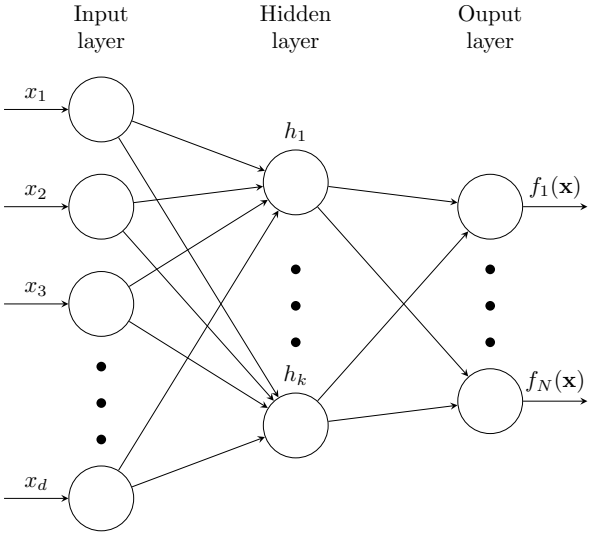

Fig. 1. A schematic of an ANN with a single fully-connected hidden layer with k nodes which takes as input $\mathbf{x} \in \mathbb{R}^d$ and produces an output $f(\mathbf{x}) \in \mathbb{R}^N$. In fully-connected configurations, each node of a given layer, aside from the output layer, is broadcast to each node of the subsequent layer, as indicated by the arrows. The value at the ith hidden layer is a scalar computed as $\sigma(\mathbf{w}_i^T\mathbf{x} + b_i)$, where $\sigma(\cdot)$ is a nonlinear activation function, $\mathbf{w}_i \in \mathbb{R}^n$ is a set of weights (i.e., coefficients), and b_i is a bias (i.e., offset).

node can then be passed to the activation function, yielding $h_i = \sigma(\mathbf{w}_i x + b_i)$. In vector form, the hidden layer is then $\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\mathbf{W}$ is a matrix with the ith row being the weights $\mathbf{w}_i$. The quantity $\mathbf{h}$ can then be sent to the output layer as another set of operations, such that

$$\mathbf{h}_2 = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 = \mathbf{W}_2\sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{b}_2 = f(\mathbf{x}) \tag{10}$$

Within this architecture, there are a number of tunable parameters, namely the weights $\mathbf{w}_i$ and biases b_i, which determine the accuracy in the approximation $f(\mathbf{x}) = y$ for some known value of y. The accuracy with which $f(\mathbf{x})$ approximates y is measured by a *loss function*. One error measure that can be used in a loss function is the mean squared error (MSE):

$$\text{MSE} = \frac{1}{n}\|\mathbf{y} - f(\mathbf{x})\|^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{11}$$

where the vectors $\mathbf{y}$ and $f(\mathbf{x})$ refer to a collection of n samples. The loss function is minimized, meaning that $f(\mathbf{x})$ is approximated as the true value y, by tuning the parameters of the ANN, a process referred to as *training*. The training procedure consists of finding the parameters $\mathbf{W}, \mathbf{b}, \mathbf{W}_2, \mathbf{b}_2$, collectively referred to as the parameters $\hat{\mathcal{P}}$, such that

$$\hat{\mathcal{P}} = \arg\min_{\mathcal{P}}\{\text{MSE}\} \tag{12}$$

In minimizing this objective function, or loss, optimization schemes such as stochastic gradient descent [7] are used, which updates parameters in a direction that causes the objective function to decrease.

Once an architecture is specified, the procedure of training an ANN then consists of a *forward pass* in which $\mathbf{x}$ is

propagated through the network to yield the estimate $f(\mathbf{x})$, followed by a *backward pass* in which gradients are computed at each layer and propagated through the network. The passing of gradients backwards through the network is referred to as *backpropagation* or *backprop* for short, and relies on the chain rule from calculus to efficiently compute gradients without performing redundant calculations. This procedure of forward and backward passes through the network is often referred to as an *epoch*, and is repeated many times, often until the loss function (i.e., MSE) converges to some specified tolerance.

D. Solving the S_N Equation using ANNs

The system from Equation (1) can be solved using an ANN. In particular, optimal network parameters $\hat{\mathcal{P}}$ that yield the lowest error in the solution to Equation (1) are found. This problem was previously considered by [4]. The method in [4] is implemented using a modern neural network framework, namely the `PyTorch` [12] deep learning framework in `Python`.

First, the network architecture in [4] is reproduced, which is a single fully-connected layer, with a hyperbolic tangent ($\tanh$) activation function leading to the output layer (i.e., $\sigma(\cdot) = \tanh(\cdot)$). Consider the set of S_N equations for the slab geometry angular flux $\boldsymbol{\Psi}(\mathbf{z}) = \boldsymbol{\Psi} \in \mathbb{R}^{d\times N}$ sampled at spatial points $\mathbf{z} \in \mathbb{R}^d$ (derived in Appendix B):

$$\nabla\boldsymbol{\Psi}\text{diag}(\boldsymbol{\mu}) + \Sigma_t\boldsymbol{\Psi} = \frac{1}{2}\big(\Sigma_{s0}\boldsymbol{\Phi}_0\mathbb{1}_N^T + 3\Sigma_{s1}\boldsymbol{\Phi}_1\boldsymbol{\mu}^T\big) + \frac{1}{2}\mathbf{Q} \tag{13}$$

where $\text{diag}(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ is a matrix with ordered values of the ordinate angles μ_i on the diagonals, and $\mathbb{1}_N^T$ is a row vector of N ones. Note that the spatial argument $\mathbf{z}$ has been dropped from the scalar and angular fluxes for brevity. The neural network is used to estimate $\hat{\boldsymbol{\Psi}}$, allowing us to compute $\hat{\boldsymbol{\Phi}}_0$ and $\hat{\boldsymbol{\Phi}}_1$ as

$$\hat{\boldsymbol{\Phi}}_0 = \hat{\boldsymbol{\Psi}}\mathbf{w} \tag{14}$$

$$\hat{\boldsymbol{\Phi}}_1 = \hat{\boldsymbol{\Psi}}\text{diag}(\boldsymbol{\mu})\mathbf{w} \tag{15}$$

where $\mathbf{w} \in \mathbb{R}^N$ are the Gauss-Legendre quadrature weights.

The error can then be measured as how well the ANN solution satisfies Equation (13) using the squared error:

$$\begin{aligned}\mathcal{L} = &\left\|\nabla\hat{\boldsymbol{\Psi}}\text{diag}(\boldsymbol{\mu}) + \Sigma_t\hat{\boldsymbol{\Psi}} - \frac{1}{2}\big(\Sigma_{s0}\hat{\boldsymbol{\Phi}}_0\mathbb{1}_N^T - 3\Sigma_{s1}\hat{\boldsymbol{\Phi}}_1\boldsymbol{\mu}^T - \mathbf{Q}\big)\right\|_F^2 \\ &+ \gamma_L\|\hat{\boldsymbol{\Psi}}^{\mu>0}(z=0) - \boldsymbol{\Psi}_L\|_F^2 \\ &+ \gamma_R\|\hat{\boldsymbol{\Psi}}^{\mu<0}(z=z_{max}) - \boldsymbol{\Psi}_R\|_F^2\end{aligned} \tag{16}$$

where F denotes the Frobenius norm. The last two terms in Equation (16) act as regularizers to enforce specific values at the boundaries (i.e., the left (L) and right (R) boundary conditions) and each $\gamma_{\{L,R\}}$ is a regularization coefficient. The training procedure consists of finding the neural network parameters such that $\hat{\boldsymbol{\Psi}}$ mininizes the loss in Equation (16).

In most ANN use cases, both the input data $\mathbf{x}$ and an associated output $y = f(\mathbf{x})$ are used in the training, a practice referred to as *supervised learning*. In this case, however, there is no labelled output y, and instead ANN parameters are found that minimize the loss accrued by performing operations on the output.

The *Adam* optimizer [8] is used to minimize the loss function in Equation (16). The Adam optimization routine updates weights $\mathbf{w}_i$ and biases b_i in a similar fashion to traditional stochastic gradient descent, however, it also adaptively updates step sizes which typically results in faster convergence than standard stochastic gradient descent.

The procedure for solving the slab geometry transport equation in Equation (13) is summarized as follows:

1) Construct the ANN and define a loss function to minimize.
2) Pass input data $\mathbf{z}$ through the network to yield $f(\mathbf{z})$.
3) Calculate the loss using Equation (16).
4) Perform backpropagation by computing the gradient of the loss with respect to the input $\mathbf{z}$ (i.e., compute $\nabla_{\mathbf{z}}\mathcal{L}$) and using this to determine the gradient with respect to each parameter in the network.
5) Use the gradients computed above to update each parameter in the network such that the parameters are updated in the direction that decreases the loss the most.
6) Repeat the forward and backwards passes until the loss converges to a specified error tolerance.
7) Using this trained network, predict the angular flux $\hat{\boldsymbol{\Psi}}$ as well as the scalar flux $\hat{\boldsymbol{\Phi}}_0$.

The convergence criterion employed for training the neural network is

$$\left|\mathcal{L}^{(l+1)} - \mathcal{L}^{(l)}\right| < \epsilon \tag{17}$$

where $\mathcal{L}^{(l)}$ and $\mathcal{L}^{(l+1)}$ are the losses computed at epoch l and $l+1$, respectively and ϵ is some small constant (e.g., 10^{-10}).

E. Solutions using the Monte Carlo Method

Unlike the S_N solution to Equation (1), which requires solving a system of coupled linear equations, the Monte Carlo (MC) solution determines particle histories by pseudo-random sampling. One disadvantage of MC is the introduction of statistical uncertainty, which decreases as $N^{\frac{1}{2}}$ where N is the number of simulation particles. Two advantages of MC over S_N are that it allows the geometry of the problem to be represented exactly by using constructive solid geometry and it allows the energy dependence to be represented exactly using continuous energy cross sections. Also, temporal and angular discretization are not necessary for MC.

All sampling employs the inverse-CDF technique, which is described here using notation, definitions, and descriptions from [10]. A random variable is a one-to-one mapping of elements from the sample space of a random experiment, S, to real numbers $X(s) = x$ where $s \in S$. The support of X is the set of real numbers $\mathcal{A} = \{x|x = X(s), s \in S\}$. A probability density function (PDF) $f(x)$ for a continuous random variable X satisfies the existence conditions $\int_{\mathcal{A}} dx f(x) = 1$ and $f(x) \geq 0$ for all real x. The cumulative density function (CDF) is $F(x) \equiv P(X \leq x) \equiv \int_{-\infty}^{x} dx f(x)$ where $f(x)$ is the PDF. Since $F(x)$ is defined as a probability, $0 \leq F(x) \leq 1$, that means $F(X)$ is uniformly distributed between 0 and 1. Random variate generation is achieved by $x \leftarrow F^{-1}(u)$ where

the inverse-CDF F^{-1} exists and $u \sim U(0,1)$ denotes a random variate that is uniformly distributed between 0 and 1. All simulation platforms come with or support the construction of high performance pseudo random number generators providing $u \sim U(0,1)$ variates.

Algorithm 1 provides an overview of the Monte Carlo algorithm used in this analysis. Note that MC is used to solve Equation (1) in a non-multiplying medium, meaning $\Sigma_f = 0$, which explains the absence of fission reactions in Algorithm 1. See Appendix D and Appendix E for derivations of the linearly anisotropic scattering outgoing angle and the distance to collision sampling algorithms, respectively.

Algorithm 1 Using Monte Carlo to determine particle histories

```
Sample N values of z and μ from Q_ext
while N_M > 0 unabsorbed particles remain do
  Compute N_M distances to cell boundary, d_{b,m}
  Sample N_M distances to collision, d_{c,m}
  if d_{b,m} < d_{c,m} then
    Move N_M − N_C particles across cell boundaries
    Accumulate path-length flux tally in cell j
  else
    Move N_C particles to collision sites
    Accumulate path-length flux tally in cell j
    Sample N_C reactions
    if reaction is absorption then
      Delete N_C − N_S absorbed particles
      Accumulate absorption reaction tally in cell j
    else
      Sample and assign N_S scattering directions
      Accumulate scattering reaction tally in cell j
    end if
  end if
end while
```

F. Model Comparison

These methods are compared by solving for the scalar flux $\phi(z)$ in a single slab of a completely absorbing material (i.e., $\Sigma_t = \Sigma_a$, $\Sigma_{s\{0,1\}} = 0$, $\Sigma_f = 0$). The slab is 1 cm in length, with vacuum boundary conditions (i.e., the angular flux coming from outside of the slab is zero) and has an external source uniformly distributed through the material. In this problem there is an exact solution for the scalar flux $\phi(z)$, shown in Fig. 2 (see Appendix C for derivation)

$$\phi(z) = \frac{Q_0}{\Sigma_a} - \frac{Q_0}{2\Sigma_a}\left(e^{-\Sigma_a z} + e^{-\Sigma_a (H-z)}\right) + \frac{Q_0}{2\Sigma_a}\left(zE_1(\Sigma_a z) + (H-z)E_1(\Sigma_a(H-z))\right) \tag{18}$$

where $z \in [0, H]$.

The three algorithms are compared using max relative error

$$r_m = \max_j \frac{|\phi_j - \hat{\phi}_j|}{\phi_j} \tag{19}$$

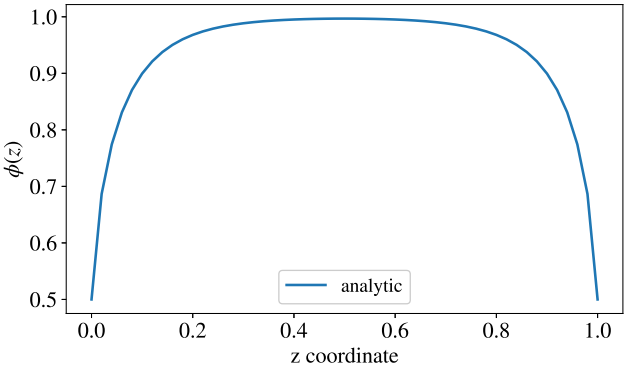

Fig. 2. Analytic solution for the scalar flux $\phi(z)$ in Problem 1, given by Equation (18).

where $m \in \{\text{nn, sn, mc}\}$ denotes the different algorithms, $\hat{\phi}_j$ is the model solution at spatial point j, and ϕ_j is the analytic solution at spatial point j.

III. Results

The implementation [1] of all three solution methods used the parameters given in Table I and was run on a 2.3 GHz Intel Core i5 2410M processor. Table II shows a comparison of the error between the various methods, as well as their runtime. From this table, one can see that the neural network achieved near-equal accuracy with discrete ordinates, but at four orders of magnitude greater runtime. Out of the three methods, Monte Carlo was the most accurate.

Fig. 3 shows the relative error as a function of position in the slab. In this figure, one can see that the neural network and discrete ordinates solutions closely match each other, and also that the maximum relative error for each algorithm is near the boundary. The "noise" in the Monte Carlo solution, resulting from the random nature of each interaction, is also visible. One other feature to note is that the neural network fails to preserve symmetry. Fig. 4 shows the neural network convergence.

After training a network using the number of zones listed in Table I, the solution was predicted at different zone counts. Fig. 5 shows the relative error as a function of number of zones. From this figure, one can see that the neural network prediction was more accurate than S_N at 10 zones. The accuracy for all other zone counts is approximately equal for both the neural network prediction and S_N.

Fig. 6 shows the time to solution for neural network prediction and S_N as a function of number of zones. This figure shows that for all zone counts except ten, the neural network prediction was almost two orders of magnitude faster than S_N. On graphics processing units (GPUs), tensor cores, and tensor processing units, neural network predictions are expected to be several more than two orders of magnitude faster than S_N for this problem.

TABLE I
PARAMETERS USED TO SPECIFY AND SOLVE THE UNIFORM SOURCE PROBLEM.

Parameter	Value	Description
Σ_t	8	Total cross section (cm^{-1})
Σ_{s0}	0	Total scattering cross section (cm^{-1})
Σ_{s1}	0	Linearly anisotropic cross section (cm^{-1})
Q_0	8	External source magnitude
J_{nn}	50	Number of zones for NN solution
J_{sn}	50	Number of zones for S_N solution
J_{mc}	50	Number of zones for MC solution
NR_{nn}	4	Number of ordinates for NN solution
NR_{sn}	4	Number of ordinates for S_N solution
NP	1e6	Number of particles for MC solution
ϵ_{nn}	1e-6	Convergence criterion value for NN solution
ϵ_{sn}	1e-13	Convergence criterion value for S_N solution
k	5	Number of hidden layer nodes in NN

TABLE II
MAX RELATIVE ERROR, ITS LOCATION, AND RUNTIMES FOR PROBLEM 1.

Algorithm	Max. rel. err.	z (cm)	Runtime (s)
nn train	-	-	6.97e+01
nn pred	0.051807	0.98	1.39e-04
sn	0.047869	0.02	4.39e-03
mc	0.013654	0.01	2.77e+00

IV. DISCUSSION

While ANNs have shown great success in tasks such as regression and classification, the use of ANNs for solving differential equations is not as mature. As a result, though ANNs are capable of solving problems such as the NTE in a slab geometry, it comes with many more design decisions (e.g., number of nodes) and heuristics (e.g., using particular values for regularization) than simply using traditional methods such as S_N or Monte Carlo. Nonetheless, using ANNs to approach problems such as the NTE is still an interesting avenue of research, and should be studied further.

In particular, some areas of interest include varying the ANN parameters (e.g., number of nodes in the hidden layer, etc.) to preserve symmetry in the solution for $\hat{\mathbf{\Phi}}$ and accelerate convergence. Additionally, other measures of accuracy can be explored to help understand how the character of the ANN solution differs from S_N and MC. Furthermore, the ANN can be run on GPUs to achieve greater speedups. Finally, the ANN can be used to solve more complicated problems, for example, a problem including scattering or a non-uniform source (e.g., distributed over half of the slab).

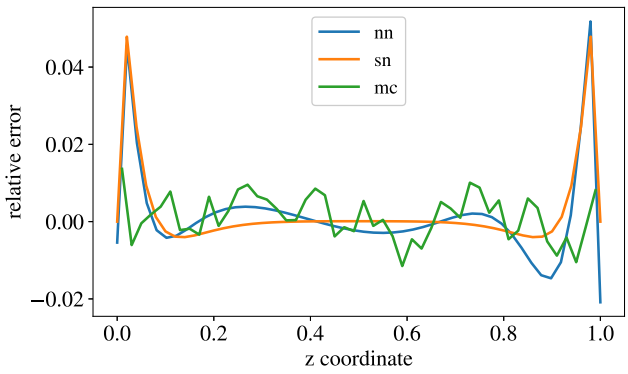

Fig. 3. Spatial distribution of relative error for Problem 1. From this, one can see that Monte Carlo (mc) is insensitive to the boundary condition, while both the neural network (nn) and discrete ordinates (sn) approaches are not, resulting in higher relative error near the edges of the slab.

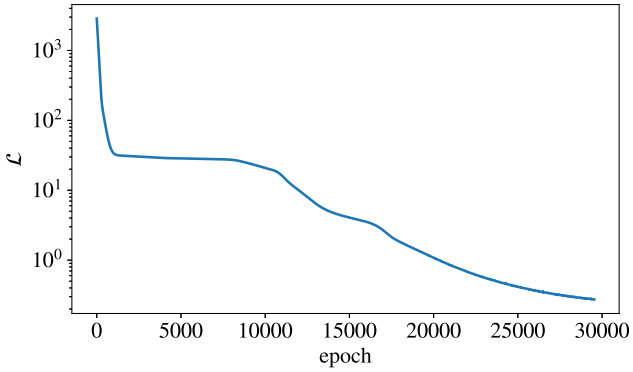

Fig. 4. Neural network loss by epoch during training for Problem 1. The loss decreases quickly over the first few hundred epochs, then remains neary flat for thousands of epochs. The flat region may be due to gradient descent getting stuck in a local minimum. Around epoch ten thousand the loss starts to decrease again, passing over a couple bumps before starting a steady decrease around epoch eighteen thousand that continues until convergence after more than thirty thousand epochs.

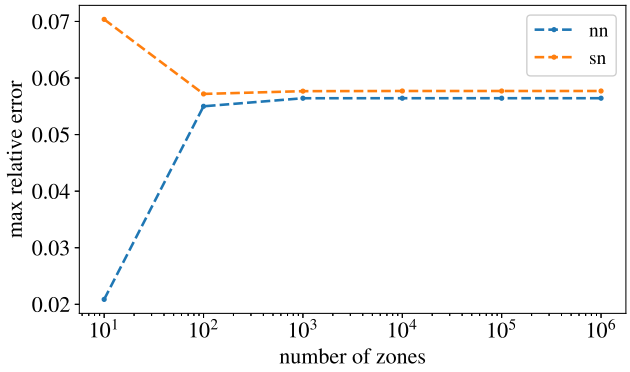

Fig. 5. Maximum relative error for predictions by zone count for Problem 1. One can see that the maximum relative error of the neural network (nn) and discrete ordinates (sn) methods is nearly equivalent for all zone counts greater than 10. This means that the two methods achieve virtually equivalent accuracy for this problem.

V. ACKNOWLEDGEMENTS

The author would like to thank Kyle Bilton for his tireless work on this project, Jasmina Vujic for asking us to work on it, and Patrick Brantley for suggesting machine learning.

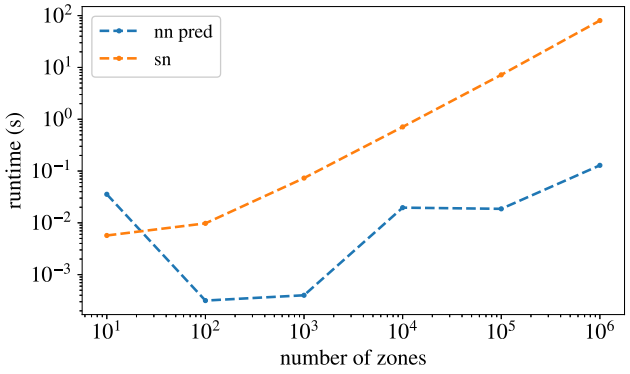

Fig. 6. Runtime by zone count for Problem 1. One can see that the runtime for all zone counts greater than 10 is two orders of magnitude less for the neural network prediction (nn pred) than it is for the discrete ordinates (sn) method. Since the solutions achieve equivalent accuracy the neural network prediction is preferred because it is much faster for this problem.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Appendix A
Slab Geometry NTE Derivation

Start with the general NTE, which has seven independent variables – three spatial variables $\mathbf{r}$, energy E, direction $\hat{\Omega}$, and time t [6]:

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\psi(\mathbf{r},E,\hat{\Omega},t)+\hat{\Omega}\cdot\nabla\psi(\mathbf{r},E,\hat{\Omega},t)+\Sigma_t(\mathbf{r},E)\psi(\mathbf{r},E,\hat{\Omega},t)\\ =\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\Sigma_s(\mathbf{r},E'\to E,\hat{\Omega}\cdot\hat{\Omega}')\psi(\mathbf{r},E,\hat{\Omega}',t)\\ +\frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\nu(E)\Sigma_f(\mathbf{r},E)\psi(\mathbf{r},E,\hat{\Omega}',t)\\ +Q_{ext}(\mathbf{r},E,\hat{\Omega},t)\end{aligned} \tag{20}$$

where $v=\sqrt{2E/m}$, Σ_t is the macroscopic total cross section, Σ_s is the macroscopic scattering cross section, $\chi(E)$ is the prompt fission neutron spectrum, $\nu(E)$ is the neutron multiplicity, Σ_f is the macroscopic fission cross section Q_{ext} is an external source, and $\psi(\mathbf{r},E,\hat{\Omega},t)$ is the angular flux.

Begin by making some simplifying assumptions:

1) This is a steady state, meaning the solution does not change with time.
2) All neutrons have equal energy and the energy does not change following a collision.
3) The external source is isotropic.
4) The fission source is isotropic.
5) Only consider one dimension in both space and angle.
6) Neutron flux does not depend on the azimuthal angle.
7) All neutrons are born as prompt.

The following definitions are used:

- $\mu\equiv\cos\theta$ is the cosine of the polar angle of the directional vector $\hat{\Omega}$
- $\mu'\equiv\cos\theta'$ is the cosine of the polar angle of the directional vector $\hat{\Omega}'$
- $\mu_0\equiv\cos\theta_0\equiv\hat{\Omega}\cdot\hat{\Omega}'$ is the cosine of the angle between the two directional vectors $\hat{\Omega}$ and $\hat{\Omega}'$

Using these assumptions and definitions, the derivation proceeds as follows [16]. First, make the following substitutions based on monoenergetic neutrons:

- $\psi(\mathbf{r},E,\hat{\Omega})=\psi(\mathbf{r},\hat{\Omega})\delta(E-E_0)$
- $\Sigma_s(\mathbf{r},E'\to E,\hat{\Omega}\cdot\hat{\Omega}')=\Sigma_s(\mathbf{r},\hat{\Omega}\cdot\hat{\Omega}')\delta(E-E_0)$
- $Q_{ext}(\mathbf{r},E,\hat{\Omega},t)=\frac{1}{4\pi}Q_{ext}(\mathbf{r})\delta(E-E_0)$

Integrate over energy E to get

$$\begin{aligned}&\int_0^\infty dE\Big[\hat{\Omega}\cdot\nabla\psi(\mathbf{r},\hat{\Omega})+\Sigma_t(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})\Big]\delta(E-E_0)\\ &=\int_0^\infty dE\delta(E-E_0)\\ &\times\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\Sigma_s(\mathbf{r},\hat{\Omega}\cdot\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')\delta(E'-E_0)\\ &+\int_0^\infty dE\Big[\frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\nu\Sigma_f(\mathbf{r})\psi(\mathbf{r},\hat{\Omega}')\delta(E'-E_0)\Big]\\ &+\frac{1}{4\pi}\int_0^\infty dEQ_{ext}(\mathbf{r})\delta(E-E_0)\end{aligned} \tag{21}$$

Now use the fact that $\chi(E)$ is a PDF, and also use the definition of scalar flux:

- $\int_0^\infty dE\chi(E)=1$
- $\int_{4\pi}d\hat{\Omega}'\psi(\mathbf{r},\hat{\Omega}')\equiv\phi(\mathbf{r})$

This yields

$$\begin{aligned}\hat{\Omega}\cdot\nabla\psi(\mathbf{r},\hat{\Omega})+\Sigma_t(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})=\int_{4\pi}d\hat{\Omega}'\Sigma_s(\mathbf{r},\hat{\Omega}\cdot\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')\\ +\frac{\nu\Sigma_f(\mathbf{r})}{4\pi}\int_{4\pi}d\hat{\Omega}'\psi(\mathbf{r},\hat{\Omega}')+\frac{1}{4\pi}Q_{ext}(\mathbf{r})\end{aligned} \tag{22}$$

Since the dot product in 3D Cartesian geometry is

$$\begin{aligned}\hat{\Omega}\cdot\nabla\psi(\mathbf{r},\hat{\Omega})&=\Omega_x\frac{\partial\psi}{\partial x}+\Omega_y\frac{\partial\psi}{\partial y}+\Omega_z\frac{\partial\psi}{\partial z}\\ &\equiv\xi\frac{\partial\psi}{\partial x}+\eta\frac{\partial\psi}{\partial y}+\mu\frac{\partial\psi}{\partial z}\end{aligned} \tag{23}$$

If an infinite and homogeneous medium in the x and y directions is assumed, then the angular flux will be a function of z only

$$\psi(\mathbf{r},\hat{\Omega})=\psi(z,\hat{\Omega}) \tag{24}$$

Then, the dot product will have only one term

$$\hat{\Omega}\cdot\nabla\psi(\mathbf{r},\hat{\Omega})\to\Omega_z\frac{\partial\psi}{\partial z}\equiv\mu\frac{\partial\psi}{\partial z} \tag{25}$$

Thus, the NTE which is time-independent, monoenergetic, and 1D in space is

$$\begin{aligned}\mu\frac{\partial\psi(z,\hat{\Omega})}{\partial z}+\Sigma_t(z)\psi(z,\hat{\Omega})=\int_{4\pi}d\hat{\Omega}'\Sigma_s(z,\hat{\Omega}\cdot\hat{\Omega}')\psi(z,\hat{\Omega}')\\ +\frac{\nu\Sigma_f(z)}{4\pi}\phi(z)+\frac{1}{4\pi}Q_{ext}(z)\end{aligned} \tag{26}$$

Recall

$$\begin{aligned}\int_{4\pi} d\hat{\Omega} &= \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta d\theta \\ &= \int_0^{2\pi} d\phi \int_1^{-1} \left[-d(\cos\theta)\right] \\ &= \int_0^{2\pi} d\phi \int_{-1}^{1} d\mu = 4\pi\end{aligned} \tag{27}$$

- $\theta \in (0,\pi) \rightarrow \cos\theta \in (1,-1)$
- $\psi(z,\hat{\Omega})d\hat{\Omega} = -\psi(z,\mu,\phi)d\mu d\phi$

Now find the relationship between the angular flux that depends on solid angle and the angular flux in the case of azimuthal symmetry

$$\begin{aligned}\int_{4\pi} d\hat{\Omega}\psi(z,\hat{\Omega}) &= \int_{-1}^{1} d\mu \int_0^{2\pi} d\phi\psi(z,\mu,\phi) \\ &= \int_{-1}^{1} d\mu\psi(z,\mu)\end{aligned} \tag{28}$$

$$\begin{aligned}\int_{-1}^{1} d\mu \int_0^{2\phi} d\phi\psi(z,\hat{\Omega}) &= \int_{-1}^{1} d\mu\psi(z,\hat{\Omega}) \int_0^{2\pi} d\phi \\ &= 2\pi \int_{-1}^{1} d\mu\psi(z,\hat{\Omega})\end{aligned} \tag{29}$$

$$\psi(z,\mu) = \frac{1}{2\pi}\psi(z,\hat{\Omega}) \tag{30}$$

By integrating the NTE over azimuthal angle,

$$\begin{aligned}&\int_0^{2\pi} d\phi \left[\mu\frac{\partial\psi(z,\hat{\Omega})}{\partial z} + \Sigma_t(z)\psi(z,\hat{\Omega})\right] = \\ &\int_0^{2\pi} d\phi \left[\int_{4\pi} d\hat{\Omega}'\Sigma_s(z,\hat{\Omega}\cdot\hat{\Omega}')\psi(z,\hat{\Omega}')\right] \\ &+ \int_0^{2\pi} d\phi \left[\frac{\nu\Sigma_f(z)}{4\pi}\phi(z) + \frac{1}{4\pi}Q_{ext}(z)\right] \\ &\mu\frac{\partial}{\partial z}\int_0^{2\pi} d\phi\psi(z,\hat{\Omega}) + \Sigma_t(z)\int_0^{2\pi} d\phi\psi(z,\hat{\Omega}) = \\ &2\pi\left[\int_{4\pi} d\hat{\Omega}'\Sigma_s(z,\hat{\Omega}\cdot\hat{\Omega}')\psi(z,\hat{\Omega}')\right] \\ &+ 2\pi\left[\frac{\nu\Sigma_f(z)}{4\pi}\phi(z) + \frac{1}{4\pi}Q_{ext}(z)\right]\end{aligned} \tag{31}$$

Which gives us the slab geometry NTE

$$\begin{aligned}\mu\frac{\partial\psi(z,\mu)}{\partial z} + \Sigma_t(z)\psi(z,\mu) &= 2\pi\int_{-1}^{1} d\mu'\Sigma_s(z,\mu_0)\psi(z,\mu') \\ &+ \frac{1}{2}\left[\nu\Sigma_f(z)\phi(z) + Q_{ext}(z)\right]\end{aligned} \tag{32}$$

Appendix B
Discrete Ordinates NTE Derivation

Start with the slab geometry NTE, which has two independent variables – one spatial variable z and one direction variable $\mu \equiv \cos\theta$:

$$\begin{aligned}&\mu\frac{\partial\psi(z,\mu)}{\partial z} + \Sigma_t(z)\psi(z,\mu) = \\ &2\pi\int_{-1}^{1} d\mu'\Sigma_s(z,\mu,\mu')\psi(z,\mu') + \frac{1}{2}Q(z)\end{aligned} \tag{33}$$

where Σ_t is the macroscopic total cross section, Σ_s is the macroscopic scattering cross section, ν is the neutron multiplicity, Q is an external source, and $\psi(z,\mu)$ is the angular flux.

Consider a domain defined by $z \in [0,H]$ and $\mu \in [-1,1]$ with vacuum boundary conditions

$$\psi(z=0,\mu) = f(\mu) = 0 \qquad 0 < \mu \leq 1 \tag{34}$$

$$\psi(z=H,\mu) = g(\mu) = 0 \qquad -1 \leq \mu < 0 \tag{35}$$

where the macroscopic scattering cross section Σ_s is expanded in a series of Legendre polynomials

$$\Sigma_s(z,\mu,\mu') = \sum_{k=0}^{N}\frac{2k+1}{4\pi}\Sigma_{sk}(z)P_k(\mu)P_k(\mu') \tag{36}$$

Define an angular quadrature set

$$\{-1 \leq \mu \leq 1\} \rightarrow \{(\mu_m, w_m) | 1 \leq m \leq N\} \tag{37}$$

$$\int_{-1}^{1} a(\mu)d\mu \rightarrow \sum_{m=1}^{N} a(\mu_m)w_m \tag{38}$$

where

- N is the order of the quadrature set
- μ_m are the discrete ordinates
- w_m are the angular weights

The derivation then proceeds as follows [9]. First, discretize in angle μ using the discrete ordinates method

$$\begin{aligned}&\mu_m\frac{\partial\psi_m(z)}{\partial z} + \Sigma_t(z)\psi_m(z) = \\ &\frac{1}{2}\sum_{n=1}^{N}\psi_n(z)w_n\Sigma_s(z,\mu_m,\mu_n) + \frac{1}{2}Q(z)\end{aligned} \tag{39}$$

with boundary conditions

$$\psi_m(z=0) = f_m \qquad \mu_m > 0 \tag{40}$$

$$\psi_m(z=H) = g_m \qquad \mu_m < 0 \tag{41}$$

In 1D, the Gauss-Legendre quadrature sets of even order N are used.

Second, discretize the discrete ordinates equations in space using the finite volume method. A spatial discretization (see Fig. 7) is chosen so that material boundaries occur at cell edges, meaning Σ_t, Σ_s, and Q are constant within cells

$$\Sigma_t(z) = \Sigma_{t,j} \tag{42}$$

$$\Sigma_s(z,\mu_m,\mu_n) = \Sigma_{s,j}(\mu_m,\mu_n) \tag{43}$$

$$Q(z) = Q_j \tag{44}$$

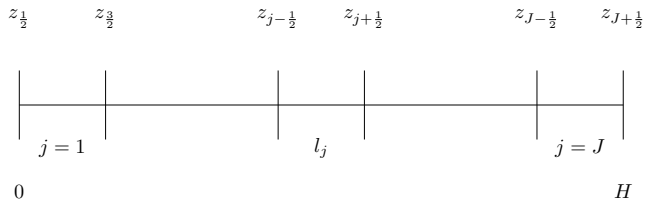

Fig. 7. Finite volume method spatial discretization for the discrete ordinates NTE in one dimensional slab geometry.

for $z_{j-\frac{1}{2}} < z < z_{j+\frac{1}{2}}$ and $1 \leq j \leq J$. Then, integrate over the jth cell

$$\int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} dz \mu_m \frac{\partial \psi_m(z)}{\partial z} + \int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} dz \Sigma_t(z) \psi_m(z) = \int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} dz \frac{1}{2} \sum_{n=1}^{N} \psi_n(z) w_n \Sigma_s(z, \mu_m, \mu_n) + \int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} dz \frac{1}{2} Q(z) \tag{45}$$

$$\frac{\mu_m}{l_j}(\psi_{m,j+\frac{1}{2}} - \psi_{m,j-\frac{1}{2}}) + \Sigma_{t,j}\psi_{m,j} = \frac{1}{2}\sum_{n=1}^{N} \Sigma_{s,j}(\mu_m, \mu_n)\psi_{n,j} w_n + \frac{1}{2} Q_j \tag{46}$$

where

$$\psi_{m,j+\frac{1}{2}} \equiv \psi_m(z_{j+\frac{1}{2}}) \tag{47}$$

$$\psi_{m,j} \equiv \frac{1}{l_j} \int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} dz \psi_m(z) \tag{48}$$

are the cell-edge angular fluxes and cell-averaged angular fluxes, respectively. Next, substitute

$$\Sigma_{s,j}(\mu_m, \mu_n) = \sum_{k=0}^{K} \Sigma_{s,j,k} P_k(\mu_m) P_k(\mu_n) \tag{49}$$

to obtain

$$\frac{\mu_m}{l_j}(\psi_{m,j+\frac{1}{2}} - \psi_{m,j-\frac{1}{2}}) + \Sigma_{t,j}\psi_{m,j} = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=0}^{K} \Sigma_{s,j,k} P_k(\mu_m) P_k(\mu_n)\psi_{n,j} w_n + \frac{1}{2} Q_j \tag{50}$$

for $1 \leq m \leq N$ and $1 \leq j \leq J$ which gives us JN equations. If boundary conditions are included,

$$\psi_{m,\frac{1}{2}} = f_m \qquad \mu_m > 0 \tag{51}$$

$$\psi_{m,J+\frac{1}{2}} = g_m \qquad \mu_m < 0 \tag{52}$$

which are N equations, there are thus $JN + N = (J+1)N$ equations. Since there are $(J+1)N$ unknown cell edge fluxes and JN unknown cell-averaged fluxes, JN more equations are needed.

Now introduce the "auxiliary" equations

$$\psi_{m,j} = \frac{1+\alpha_{m,j}}{2}\psi_{m,j+\frac{1}{2}} + \frac{1-\alpha_{m,j}}{2}\psi_{m,j+\frac{1}{2}} \tag{53}$$

for $1 \leq m \leq N$ and $1 \leq j \leq J$ which gives us JN more equations. Choose the diamond difference relation, $\alpha_{m,j} = 0$, which yields the arithmetic mean of the edges.

The spatially-discretized discrete ordinates equations can be simplified by setting $K = 1$ for linearly anisotropic scattering

$$\sum_{n=1}^{N}\sum_{k=0}^{K} \Sigma_{s,j,k} P_k(\mu_m) P_k(\mu_n)\psi_{n,j} w_n = \Sigma_{s0,j}\Phi_{0,j} + 3\mu_m \Sigma_{s1,j}\Phi_{1,j} \tag{54}$$

where

$$\Phi_{0,j} \equiv \sum_{m=1}^{N} \psi_{m,j} w_m \tag{55}$$

$$\Phi_{1,j} \equiv \sum_{m=1}^{N} \mu_m \psi_{m,j} w_m \tag{56}$$

are the cell-averaged scalar flux and the cell-averaged current, respectively.

Substitute into spatially-discretized discrete ordinates equations to arrive at linearly anisotropic approximation equations

$$\frac{\mu_m}{l_j}(\psi_{m,j+\frac{1}{2}} - \psi_{m,j-\frac{1}{2}}) + \Sigma_{t,j}\psi_{m,j} = \frac{1}{2}\left[\Sigma_{s0,j}\Phi_{0,j} + 3\mu_m\Sigma_{s1,j}\Phi_{1,j} + Q_{m,j}\right] \tag{57}$$

$$\psi_{m,j} = \frac{1+\alpha_{m,j}}{2}\psi_{m,j+\frac{1}{2}} + \frac{1-\alpha_{m,j}}{2}\psi_{m,j+\frac{1}{2}} \tag{58}$$

$$\psi_{m,\frac{1}{2}} = f_m \qquad \mu_m > 0 \tag{59}$$

$$\psi_{m,J+\frac{1}{2}} = g_m \qquad \mu_m < 0 \tag{60}$$

where

$$\Phi_{0,j} \equiv \sum_{m=1}^{N} \psi_{m,j} w_m \tag{61}$$

$$\Phi_{1,j} \equiv \sum_{m=1}^{N} \mu_m \psi_{m,j} w_m \tag{62}$$

Finally, define the source iteration process

$$\frac{\mu_m}{l_j}\left(\psi^{(l+1)}_{m,j+\frac{1}{2}} - \psi^{(l+1)}_{m,j-\frac{1}{2}}\right) + \Sigma_{t,j}\psi^{(l+1)}_{m,j} = \frac{1}{2}\hat{Q}^{(l)}_{m,j} \tag{63}$$

where

$$\hat{Q}^{(l)}_{m,j} = \Sigma_{s0,j}\Phi^{(l)}_{0,j} + 3\mu_m\Sigma_{s1,j}\Phi^{(l)}_{1,j} + Q_{m,j} \tag{64}$$

$$\Phi^{(l+1)}_{0,j} \equiv \sum_{m=1}^{N} \psi^{(l+1)}_{m,j} w_m \tag{65}$$

$$\Phi^{(l+1)}_{1,j} \equiv \sum_{m=1}^{N} \mu_m \psi^{(l+1)}_{m,j} w_m \tag{66}$$

$$\psi^{(l+1)}_{m,j} = \frac{1+\alpha_{m,j}}{2}\psi^{(l+1)}_{m,j+\frac{1}{2}} + \frac{1-\alpha_{m,j}}{2}\psi^{(l+1)}_{m,j+\frac{1}{2}} \tag{67}$$

with boundary conditions

$$\psi^{(l+1)}_{m,\frac{1}{2}} = f_m \qquad \mu_m > 0 \tag{68}$$

$$\psi^{(l+1)}_{m,J+\frac{1}{2}} = g_m \qquad \mu_m < 0 \tag{69}$$

APPENDIX C
PROBLEM 1 ANALYTIC DERIVATION

Start with the purely-absorbing medium slab geometry NTE with constant absorption cross section and external source. There are two independent variables – one spatial variable z and one direction variable $\mu \equiv \cos\theta$:

$$\mu\frac{\partial\psi(z,\mu)}{\partial z} + \Sigma_a\psi(z,\mu) = \frac{Q_0}{2} \tag{70}$$

where Σ_a is the macroscopic absorption cross section, Q_0 is an external source, and $\psi(z,\mu)$ is the angular flux.

Consider a domain defined by $z \in [0,H]$ and $\mu \in [-1,1]$ with vacuum boundary conditions

$$\psi(z=0,\mu) = f(\mu) = 0 \qquad 0 < \mu \leq 1 \tag{71}$$
$$\psi(z=H,\mu) = g(\mu) = 0 \qquad -1 \leq \mu < 0 \tag{72}$$

The analytic solution for the angular flux is [16]

$$\psi(z,\mu) = \begin{cases} \frac{Q_0}{2\Sigma_a}\left[1 - e^{-\frac{\Sigma_a}{\mu}z}\right], & 0 < \mu < 1, z \in [0,H] \\ \frac{Q_0}{2\Sigma_a}\left[1 - e^{-\frac{\Sigma_a}{|\mu|}(H-z)}\right], & -1 < \mu < 0, z \in [0,H] \end{cases} \tag{73}$$

The scalar flux is

$$\begin{aligned}\phi(z) &= \int_{-1}^{1} d\mu\, \psi(z,\mu) \\ &= \frac{Q_0}{2\Sigma_a}\int_0^1 d\mu\left(1 - e^{-\frac{\Sigma_a}{\mu}z}\right) \\ &+ \frac{Q_0}{2\Sigma_a}\int_{-1}^0 d\mu\left(1 - e^{-\frac{\Sigma_a}{|\mu|}(H-z)}\right)\end{aligned} \tag{74}$$

For the first integral, $\phi_a(z)$, let $u = \frac{1}{\mu}$, $du = -\frac{1}{u^2}d\mu$, then

$$\begin{aligned}\phi_a(z) &= \frac{Q_0}{2\Sigma_a}\int_\infty^1 \frac{-du}{u^2}\left(1 - e^{-\Sigma_a z u}\right) \\ &= \frac{Q_0}{2\Sigma_a}\left(\frac{1}{u}\Big|_\infty^1 - \frac{e^{-\Sigma_a z u}}{u}\Big|_\infty^1 - \Sigma_a z\int_\infty^1 du\frac{e^{-\Sigma_a z u}}{u}\right) \\ &= \frac{Q_0}{2\Sigma_a}\left(1 - e^{-\Sigma_a z} + \Sigma_a z\int_1^\infty du\frac{e^{-\Sigma_a z u}}{u}\right) \\ &= \frac{Q_0}{2\Sigma_a}\left(1 - e^{-\Sigma_a z} + \Sigma_a z E_1(\Sigma_a z)\right)\end{aligned} \tag{75}$$

where $E_1(\Sigma_a z)$ is the exponential integral. For the second integral, $\phi_b(z)$, let $u = -\frac{1}{\mu}$, $du = \frac{1}{u^2}d\mu$, then

$$\begin{aligned}\phi_b(z) &= \frac{Q_0}{2\Sigma_a}\int_1^\infty \frac{du}{u^2}\left(1 - e^{-\Sigma_a(H-z)u}\right) \\ &= \frac{Q_0}{2\Sigma_a}\left(-\frac{1}{u}\Big|_1^\infty + \frac{e^{-\Sigma_a(H-z)u}}{u}\Big|_1^\infty\right. \\ &\left. + \Sigma_a(H-z)\int_1^\infty du\frac{e^{-\Sigma_a(H-z)u}}{u}\right) \\ &= \frac{Q_0}{2\Sigma_a}\left(1 - e^{-\Sigma_a(H-z)} + \Sigma_a(H-z)E_1(\Sigma_a(H-z))\right)\end{aligned} \tag{76}$$

Thus, the analytic solution is

$$\begin{aligned}\phi(z) =& \frac{Q_0}{\Sigma_a} - \frac{Q_0}{2\Sigma_a}\left(e^{-\Sigma_a z} + e^{-\Sigma_a(H-z)}\right) \\ &+ \frac{Q_0}{2\Sigma_a}\left(zE_1(\Sigma_a z) + (H-z)E_1(\Sigma_a(H-z))\right)\end{aligned} \tag{77}$$

APPENDIX D
LINEARLY ANISOTROPIC SAMPLING

The objective is to sample the new direction of a scattered neutron in the case of linearly anisotropic scattering. Our derivation [16] starts with the slab geometry NTE

$$\begin{aligned}\mu\frac{d\psi(z,\mu)}{dz} + \Sigma_t(z)\psi(z,\mu) =& \\ 2\pi\int_{-1}^1 d\mu'\Sigma_s(z,\mu_0)\psi(z,\mu') &+ \frac{1}{2}Q_{ext}(z)\end{aligned} \tag{78}$$

The in-scattering angular neutron source is

$$S(z,\mu) = 2\pi\int_{-1}^1 d\mu'\Sigma_s(z,\mu_0)\psi(z,\mu') \tag{79}$$

The double differential scattering cross section is expanded into Legendre polynomials

$$\Sigma_s(z,\mu_0) = \sum_{l=0}^\infty \frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu_0) \tag{80}$$

For the linearly anisotropic case, take the first two terms in the expansion

$$\begin{aligned}\Sigma_s(z,\mu_0) &\approx \sum_{l=0}^1 \frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu_0) \\ &= \frac{1}{2\pi}\left[\frac{1}{2}\Sigma_{s0}(z)P_0(\mu_0) + \frac{3}{2}\Sigma_{s1}(z)P_1(\mu_0)\right] \\ &= \frac{1}{2\pi}\left[\frac{1}{2}\Sigma_{s0}(z) + \frac{3}{2}\mu_0\Sigma_{s1}(z)\right]\end{aligned} \tag{81}$$

where $\mu_0 \equiv \mu \times \mu'$ is the cosine between the incoming and outgoing directional vectors in 1D (μ' and μ, respectively). Define $\overline{\mu}_0 = \overline{\cos\theta_0} = \frac{\Sigma_{s1}(z)}{\Sigma_{s0}(z)}$ and substitude to get

$$\Sigma_s(z,\mu_0) \approx \frac{\Sigma_{s0}(z)}{2\pi}\left[\frac{1}{2} + \frac{3}{2}\mu\mu'\overline{\mu}_0\right] \tag{82}$$

which allows us to write the in-scattering angular neutron source as

$$S(z,\mu) = \frac{\Sigma_{s0}(z)}{2}\int_{-1}^1 d\mu'\left[1 + 3\mu\mu'\overline{\mu}_0\psi(z,\mu')\right] \tag{83}$$

The PDF for linearly anisotropic scattering is

$$f(\mu) = \frac{\Sigma_s(z,\mu)}{\Sigma_s(z)} = \frac{\Sigma_s(z,\mu)}{\Sigma_{s0}(z)} = \frac{1}{2}\left[1 + 3\mu\mu_n\overline{\mu}_0\right] \tag{84}$$

The CDF is

$$\begin{aligned} F(\mu) &= \int_{-1}^{\mu} d\mu'\, f(\mu') \\ &= \frac{1}{2}\int_{-1}^{\mu} d\mu' \left[1 + 3\mu\mu_n\overline{\mu}_0\right] \\ &= \frac{1}{2}\left[(\mu+1) + 3\mu_n\overline{\mu}_0\frac{1}{2}(\mu^2-1)\right] \end{aligned} \tag{85}$$

Let ξ be a random variate generated from a $U(0,1)$ random number generator. Use the inverse-CDF technique

$$\begin{aligned} F(\mu) = \frac{1}{2}\left[(\mu+1) + 3\mu_n\overline{\mu}_0\frac{1}{2}(\mu^2-1)\right] &= \xi \\ \left(\frac{3}{2}\mu_n\overline{\mu}_0\right)\mu^2 + \mu - \left(2\xi - 1 + \frac{3}{2}\mu_n\overline{\mu}_0\right) &= 0 \\ \mu = \frac{-1 \pm \sqrt{1-(3\mu_n\overline{\mu}_0)[2(1-2\xi)-3\mu_n\overline{\mu}_0]}}{3\mu_n\overline{\mu}_0} & \end{aligned} \tag{86}$$

Choose the "+" so that $-1 \leq \mu \leq 1$

$$\begin{aligned} \mu &= \frac{-1+\sqrt{1-(3\mu_n\overline{\mu}_0)[2(1-2\xi)-3\mu_n\overline{\mu}_0]}}{3\mu_n\overline{\mu}_0} \\ \mu &= \frac{-1+\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}}{3\mu_n\overline{\mu}_0} \\ &\times \frac{-1-\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}}{-1-\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}} \\ \mu &= \frac{1-[1-(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]]}{-3\mu_n\overline{\mu}_0 \times [1+\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}]} \\ \mu &= \frac{2(2\xi-1)+3\mu_n\overline{\mu}_0}{1+\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}} \end{aligned} \tag{87}$$

Thus, the outgoing scattering direction (n+1) can be sampled as

$$\mu = \frac{2(2\xi-1)+3\mu_n\overline{\mu}_0}{1+\sqrt{1+(3\mu_n\overline{\mu}_0)[2(2\xi-1)+3\mu_n\overline{\mu}_0]}} \tag{88}$$

Note that if scattering is isotropic, $\overline{\mu}_0 = 0$ and $\mu_{n+1} = 2\xi - 1$.

Appendix E
Distance to Collision Sampling

Assume that the distances traveled by neutrons arriving at collision sites may be modeled as a Poisson process. Also assume that the total macroscopic cross section Σ_t is constant within the zone of consideration. The probability that a neutron travels a distance r between collisions is then

$$f(r) = \Sigma_t e^{-\Sigma_t r} \tag{89}$$

The inverse-CDF technique to sample distances is used, which first requires computing the CDF of Equation (89)

$$\begin{aligned} F(r) = \int_0^r dr\, f(r) &= \int_0^r dr\, \Sigma_t e^{-\Sigma_t r} \\ &= -e^{-\Sigma_t r}\Big|_0^r \\ &= -(e^{-\Sigma_t r} - 1) \\ &= 1 - e^{-\Sigma_t r} \end{aligned} \tag{90}$$

$F(r)$ can then be solved for r

$$\begin{aligned} F(r) &= 1 - e^{-\Sigma_t r} \\ 1 - F(r) &= -e^{-\Sigma_t r} \\ \ln(1-F(r)) &= -\Sigma_t r \\ \frac{-\ln(1-F(r))}{\Sigma_t} &= r \end{aligned} \tag{91}$$

Since $0 \leq F(r) \leq 1$, a random variate ξ_i generated from a $U(0,1)$ random number generator can then be substituted for $F(r)$, allowing one to sample $r_i \in [0,\infty]$ using

$$\begin{aligned} r_i &= \frac{-\ln(1-\xi_i)}{\Sigma_t} \\ r_i &= \frac{-\ln(\xi_i)}{\Sigma_t} \end{aligned} \tag{92}$$

where $1-\xi_1$ is equivalent to ξ_i because $\xi \sim U(0,1)$.

References

[1] Machine learning for neutron transport github page. https://github.com/kjbilton/neutron_transport_net. Accessed: 2018-12-08.

[2] Nvidia tensor cores. https://www.nvidia.com/en-us/data-center/tensorcore. Accessed: 2018-12-08.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Patrick S. Brantley. Spatial treatment of the slab-geometry discrete ordinates equations using artificial neural networks. Technical Report UCRL-JC-143205, Lawrence Livermore National Laboratory, Livermore, California, September 2001.

[5] M. M. Chiaramonte and M. Kiener. Solving differential equations using neural networks. http://cs229.stanford.edu/proj2013/ChiaramonteKiener-SolvingDifferentialEquationsUsingNeuralNetworks.pdf, December 2013. Accessed: 2018-12-08.

[6] James J. Duderstadt and Louis J. Hamilton. *Nuclear reactor analysis*. Wiley, 1976.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[8] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

[9] Ed Larsen. Personal communication.

[10] Lawrence M. Leemis. *Probability*. Lightning Source, 2011. http://www.math.wm.edu/~leemis.

[11] E. E. Lewis and Jr. Miller, W. F. *Computational methods of neutron transport*. Wiley, 1984.

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. Technical report, Brown University, University of Pennsylvania, November 2017.

[15] Kaz Sato, Cliff Young, and David Patterson. An in-depth look at googles first tensor processing unit (tpu). https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu, May 2017. Accessed: 2018-12-08.

[16] Jasmina Vujic. Personal communication.